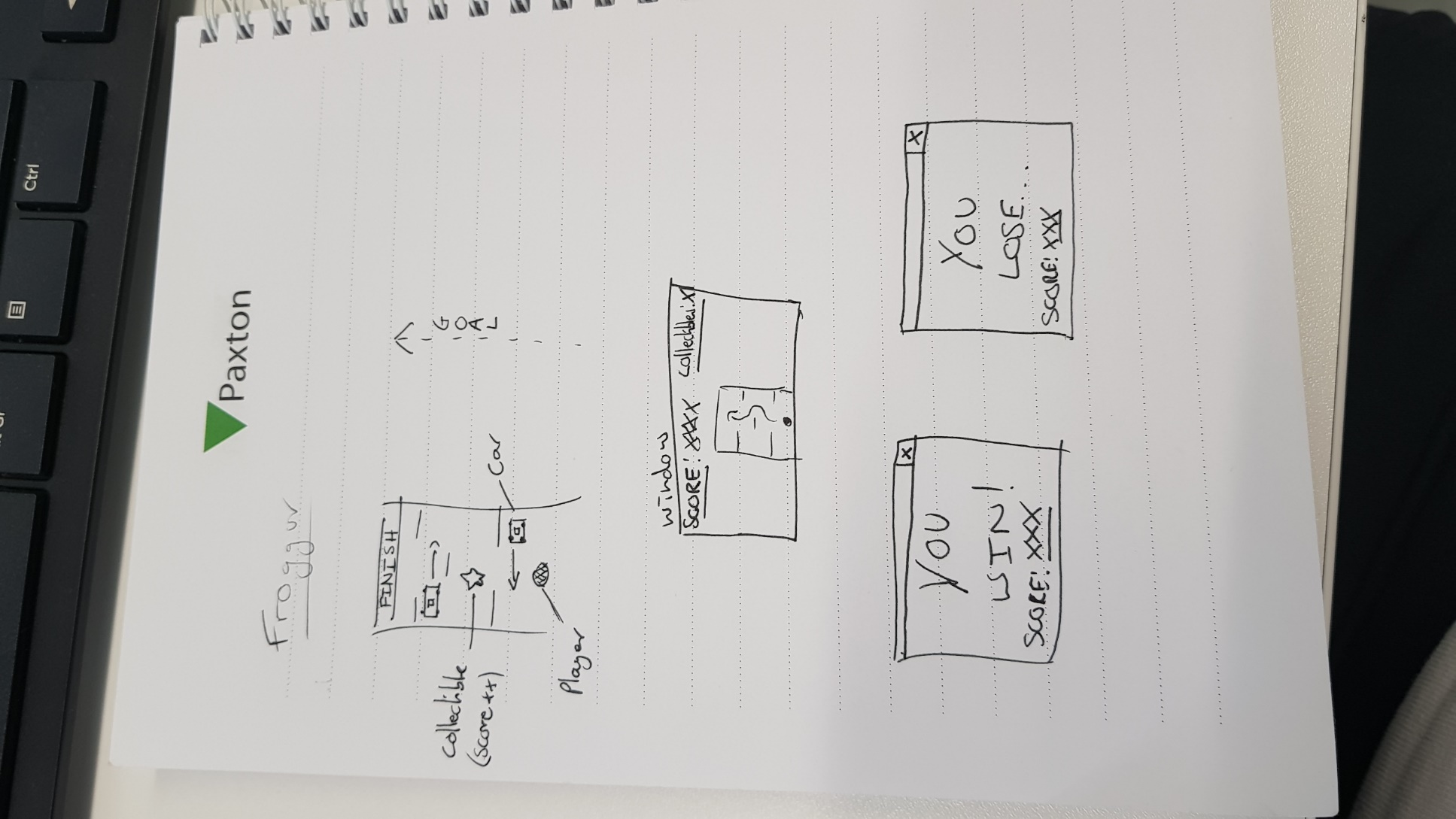
CI224 Game Development – Froggur

# Game Concept

Our game concept is effectively a clone of the arcade classic Froggur. The goal of the game is for the player to move from one side of the map (a road) to the other, avoiding obstacles and collecting collectables. If the player reaches the other side of the road without dying, they win the game. The player will also attempt to set a high score by collecting as many of the collectables placed across the road as possible.



The target platform for the game is primarily a desktop PC or laptop, although mobile and consoles would lend themselves to the game equally well.

# Core mechanics and User Interface

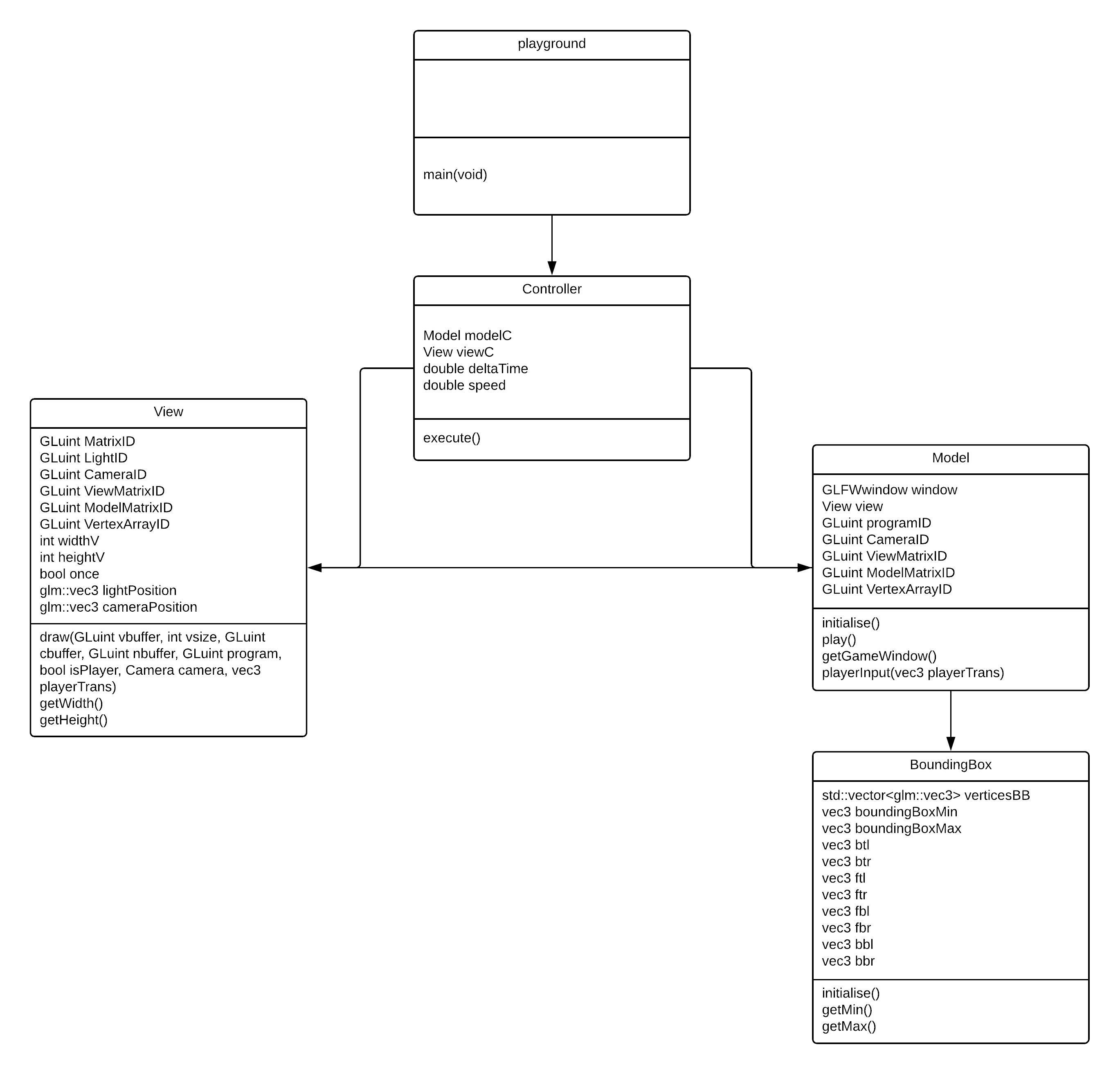
Currently there is no UI as we could not implement text to screen due to an unknown error.

Core mechanics include:

* Up Down Left Right movement.
* Collision with obstacles
* Start and finish line (No win or lose status)
* High scores (not implemented yet)

# Implementation

We began by taking the code-base from the tutorials and repurposing it in order to create a program which adhered to the MVC design pattern, creating classes for the game model, view and controller. This posed some challenges as knowing how to separate out all of the code from the main playground.cpp file into three separate classes wasn’t entirely clear at first. However we overcame the challenge and in the end produced some well organised code. The following UML Class Diagram documents the structure of the program.



The next challenge was loading a model into our scene and displaying it in the window. In the end, shaders became vitally important in making the game look and feel, like a game. We originally had some issues with the simple vertex and fragment shaders, which were shortly resolved after some troubleshooting with Karina. Advancing these shaders however proved a challenge, but with a great deal perseverance looking through open-gl tutorial sites such as <https://learnopengl.com/> and <http://www.opengl-tutorial.org/> we finally gained an understanding of how shaders worked, and ended up with an implementation of Ambient, Diffuse and Specular lighting use throughout the scene.

When it comes to the models, we decided to use fairly simple models in the implementation of the game, however we did source a royalty free low-poly asset [[1]](#footnote-1)for a car to enhance immersion. The first problem we encountered when spawning in models was the angle at which they spawned relative to the camera. It took a lot of fiddling however in the end we managed to align the camera so that the map was in view all the time. Once we had spawned the map and the player using the shaders mentioned above, the next point of call was creating our vehicles using the car asset retrieved online, and making it move across the map. This implementation was simplified as at the same time we had been working on creating the controls for the player, which had given us some insight into how to translate geometries in open-gl.

After we had most of the game objects spawning in and moving about, the next point of call was detecting collision. First we started by following the tutorials and reading in the vertices of the models to determine their minimum and maximum x values. Then, we used these calculated values to create the vertices of the bounding boxes. Once we did that it was all a matter of comparisons of bounding boxes to achieve collision. We added a ‘boop’ effect to push the player back if it collides with something. Currently handling the collision for so many cars lags the system a lot, commenting these out would be a good idea. Also it is not clear on the university computers whether the collisions are actually handled between cars as the system lags so much there is a very delayed response if any.

Overall I would say that we’ve all learnt a lot about open-gl, through perseverance and the will to prove the other teams wrong, that this assignment was indeed possible.

For ease of team-working, we have also been using github for source management. The git repository can be found here: <https://github.com/SebRabau/Froggur-Open-GL>

1. <https://www.turbosquid.com/3d-models/3d-city-vehicles-simple-model-1160672> [↑](#footnote-ref-1)